**CST-305: Project 2 – Runge-Kutta-Fehlberg (RKF) for ODE**

**Jordan Scott & Angel Velazquez**

**Responsibilities:**

Both members were focused on the program, whereas Jordan focused on doing the RKF method by hand and Angel verified the work by using the program and completing the README file. Both members also did the documentation and review the code.

**System Performance Context:**

The program runs and shows the first 5 solutions, then the program shows the first 1,000 to 2,000 solutions through a graph that the program plots the data on.

**Specific Problem:**

The specific problem solved is creating a Python program to solve an ODE for the function and doing the math by hand, afterward verifying the answer by using the ODE on the Python program and matching the results. Then the program is also supposed to show not just solutions for the first 5 solutions, but up to 1000 to 2000 depending on the hardware that the program is running on.

**Mathematical Approach:**

The approach uses the Runge-Kutta Fehlberg method. This method consists of the following.

You are given and h.

To find , we need to use where:

And

To find the rest of the values of x and y we would use the following:

**Approach for implementation in code:**

The approach for implementing this in Python was importing the formula of the ODE into Python code. The **t4(k1, k2, k3, k4)**: This function calculates the weighted sum of the four intermediate slopes obtained in the fourth-order Runge-Kutta method. It takes four parameters **k1**, **k2**, **k3**, and **k4**, which represent the slopes at different intermediate points, and returns their weighted average. We can do this by defining a function ‘dydx’ and ‘dydx2’ that has parameters that take in a x and y value and have a variable dydx that shows the ‘dydx’ function we are solving for .Next we would define our RK4 function title “rungekutta”. This will take in our initial x(x0), and y(y0) and our h. This will return the y value. Next we would implement our recursive. **plot\_recursive(x, y, count)**: This recursive function computes and plots points along the solution curve using the Runge-Kutta method until a certain condition is met. It takes three parameters: **x** and **y**, representing the current values of the independent and dependent variables, and **count**, representing the recursion depth. It does not return any value but prints intermediate results and stores the computed points in global lists.

Afterwards, we need to define the initial x to equal 1and y to equal 5 and respectively along with the step-size(h) being 0.02. The code begins by initializing two empty lists, **x\_values** and **y\_values**, which are intended to store the calculated points along the solution curve. Following this, it starts timing the computation to measure its duration. The **plot\_recursive** function is then invoked with the initial conditions to recursively compute and plot points along the solution curve using the Runge-Kutta method. Once the recursion completes, the code calculates and prints both the computational time taken and the number of computational steps involved in the process. Additionally, a range of x-values, denoted as **x\_odeint**, is generated to represent the independent variable for the true solution obtained using the **odeint** function.

The code proceeds by configuring a subplot layout with a 1x3 arrangement to facilitate the plotting of the Runge-Kutta solution, the true solution derived from `odeint`, and both solutions simultaneously. Subsequently, it plots the Runge-Kutta solution, the true solution, and both solutions on their respective subplots. Finally, the code displays the plot to visualize and compare the solutions obtained from the Runge-Kutta method and the `odeint` function.

The ODE is .

|  |  |  |  |
| --- | --- | --- | --- |
| Method: RUNGE-KUTTA METHOD | | | |
| Problem: | | | |
|  |  | | True Solution |
|  |  |
|  |  |  | 5 |
|  | 1.02 | 5.057619554621473 | 5.057619551 |
|  | 1.04 | 5.114098165612231 | 5.114098171 |
|  | 1.06 | 5.169458425170396 | 5.169458431 |
|  | 1.08 | 5.22372247813853 | 5.223722481 |
|  | 1.1 | 5.276912030861883 | 5.276912031 |

**Screenshots:**

Graph of the RK4 solution: A graph with red lines

Description automatically generated

Graph of the ODE solution:

A graph with a blue line

Description automatically generated

Both RK4 and ODE solutions together:

A graph with a blue line

Description automatically generated

Over 1,000 points for ODE and RK4

A screenshot of a computer

Description automatically generated

Console displaying the elapsed time from the program:

A black background with white numbers

Description automatically generated

**Github Repo:** [Click Here](https://github.com/angel-vlzqz/Modeling-and-Simulation/tree/main/projects/project%202%20CLC)